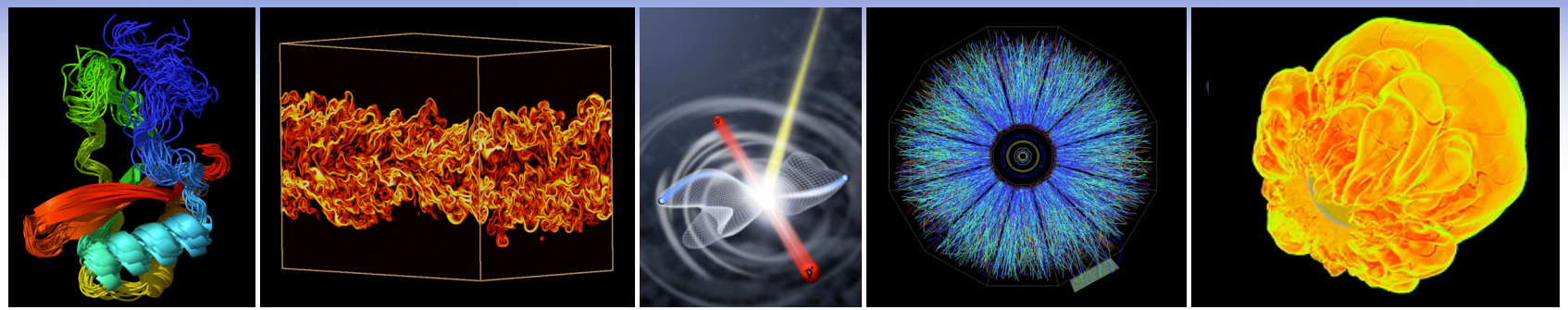


Lustre Performance From the User Perspective

HDF5 Workshop

Katie Antypas

January 20, 2009

Office of Science
U.S. DEPARTMENT OF ENERGY

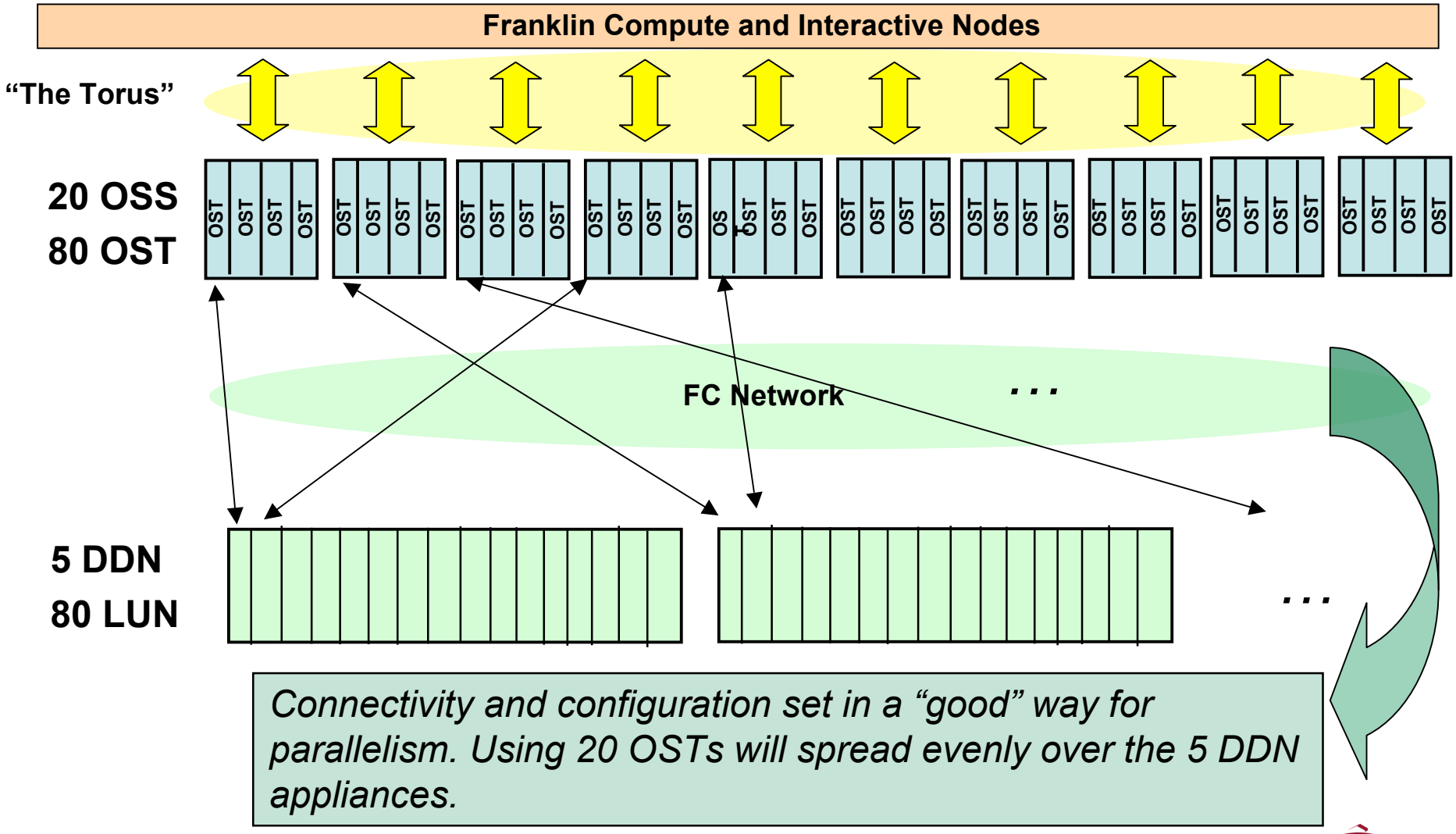
NERSC

BERKELEY LAB

User I/O Wish List

- Single shared file for parallel I/O
- Higher level portable file format
- Consistent I/O performance over a broad range of patterns (within reason)
- Shared file performance matches (or is close to) one file-per-proc performance
- No worries about file striping

Franklin Configuration

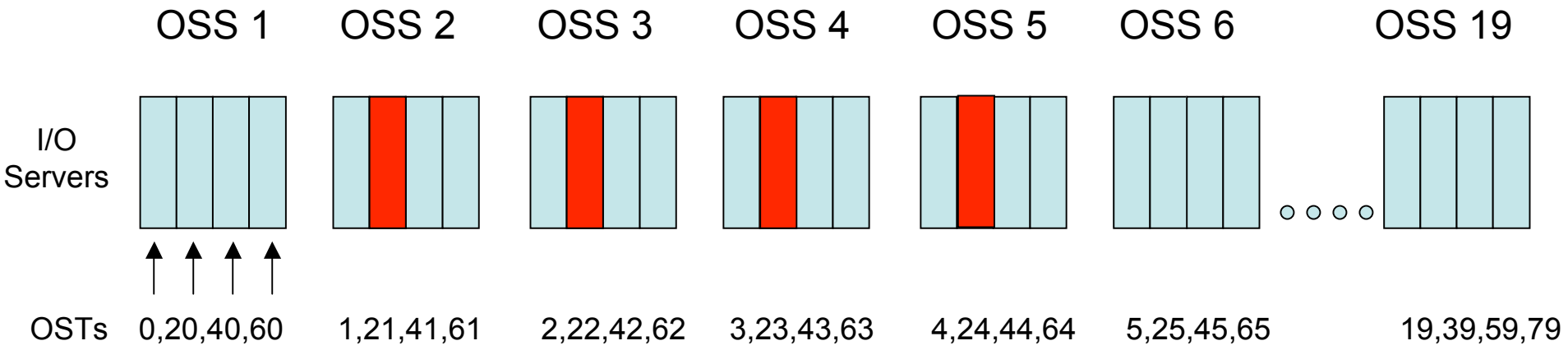


Connectivity and configuration set in a "good" way for parallelism. Using 20 OSTs will spread evenly over the 5 DDN appliances.

File Striping on Lustre

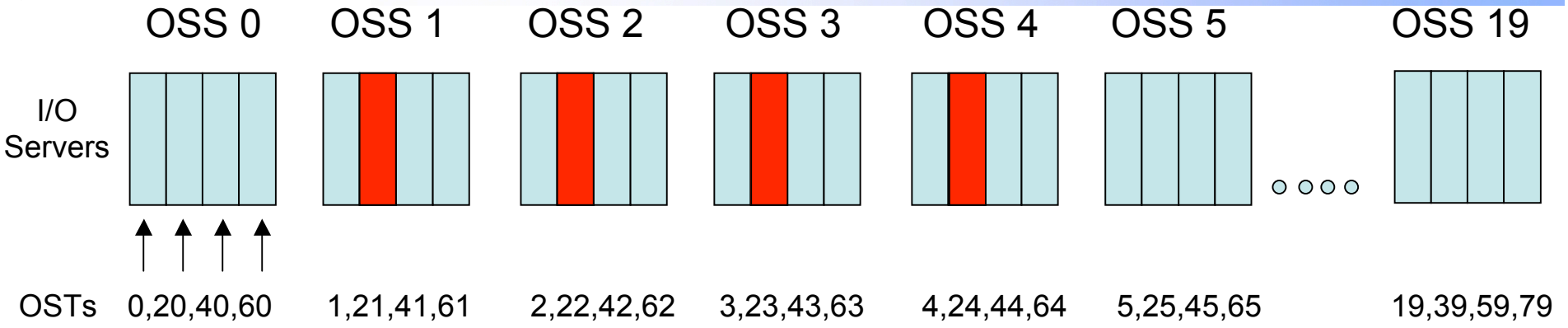
- Lustre file system on Franklin made up of an underlying set of parallel I/O servers
  - OSSs (Object Storage Servers) - nodes dedicated to I/O connected to high speed torus interconect
  - OSTs (Object Storage Targets) software abstraction of physical disk (1 OST maps to 1 LUN)
- File is said to be striped when read and write operations access multiple OSTs concurrently
- Striping can increase I/O performance since writing or reading from multiple OSTs simultaneously increases the available I/O bandwidth

Default Stripe Count of 4 on /scratch



- **Advantages**
  - Get 4 times the bandwidth you could from using 1 OST
  - Max bandwidth to 1 OST ~ 350 MB/Sec
  - Using 4 OSTs ~1,400 MB/Sec
  - In practice using all OSTs 11-12 GB/Sec

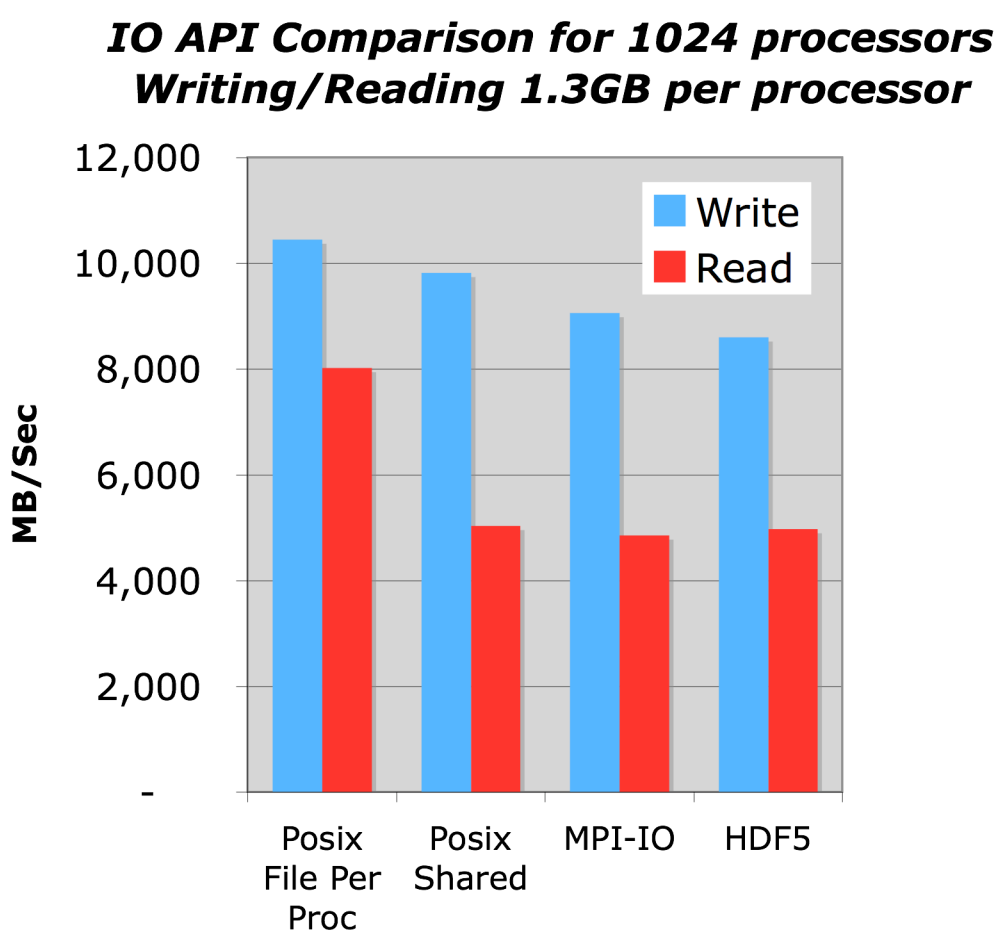
Default Striping on /scratch



- **2 other parameters which characterize striping pattern of a file**
  - **Stripe size**
    - **Number of bytes to write on each OST before cycling to next OST**
    - **Default is 1MB**
  - **OST offset**
    - **Indicates starting OST**
    - **Default is round robin across all requests on system**

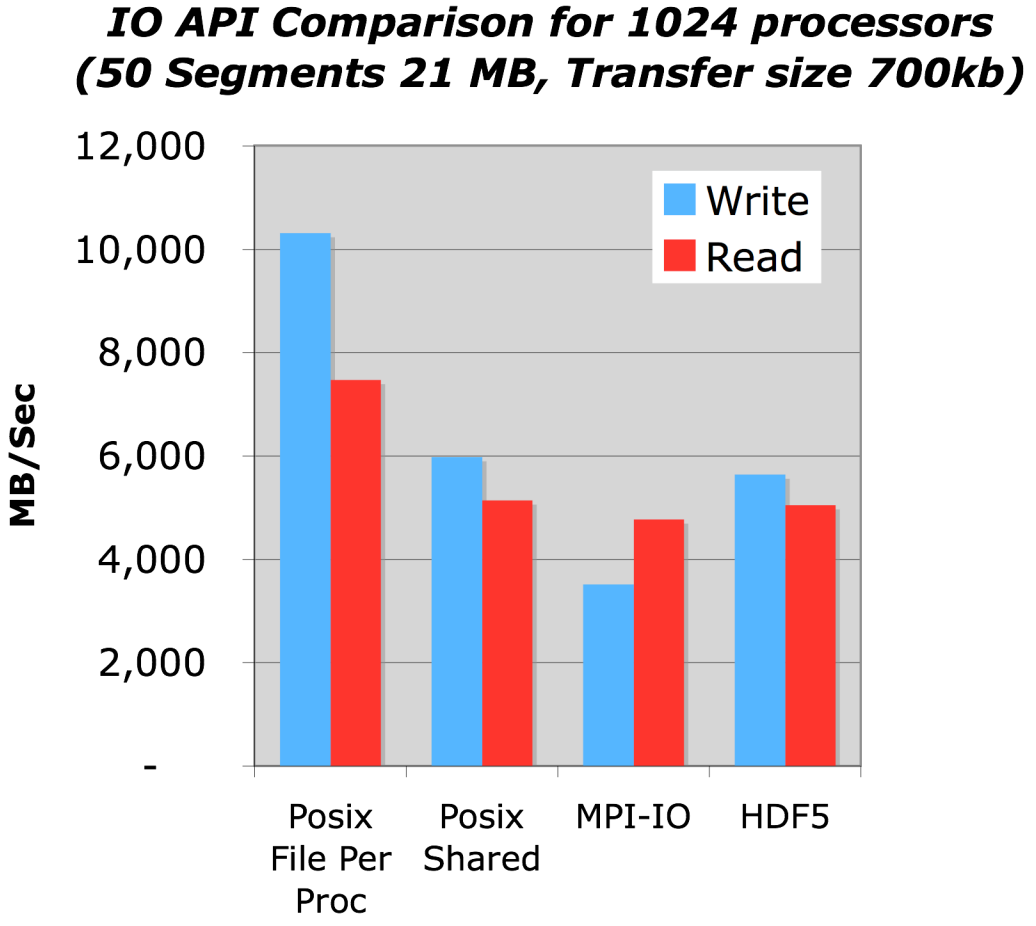
Good I/O Performance with Simple I/O Patterns

- File system capable of high performance for shared files
- Large block sequential I/O
- Transfer size multiple of stripe size
- No metadata



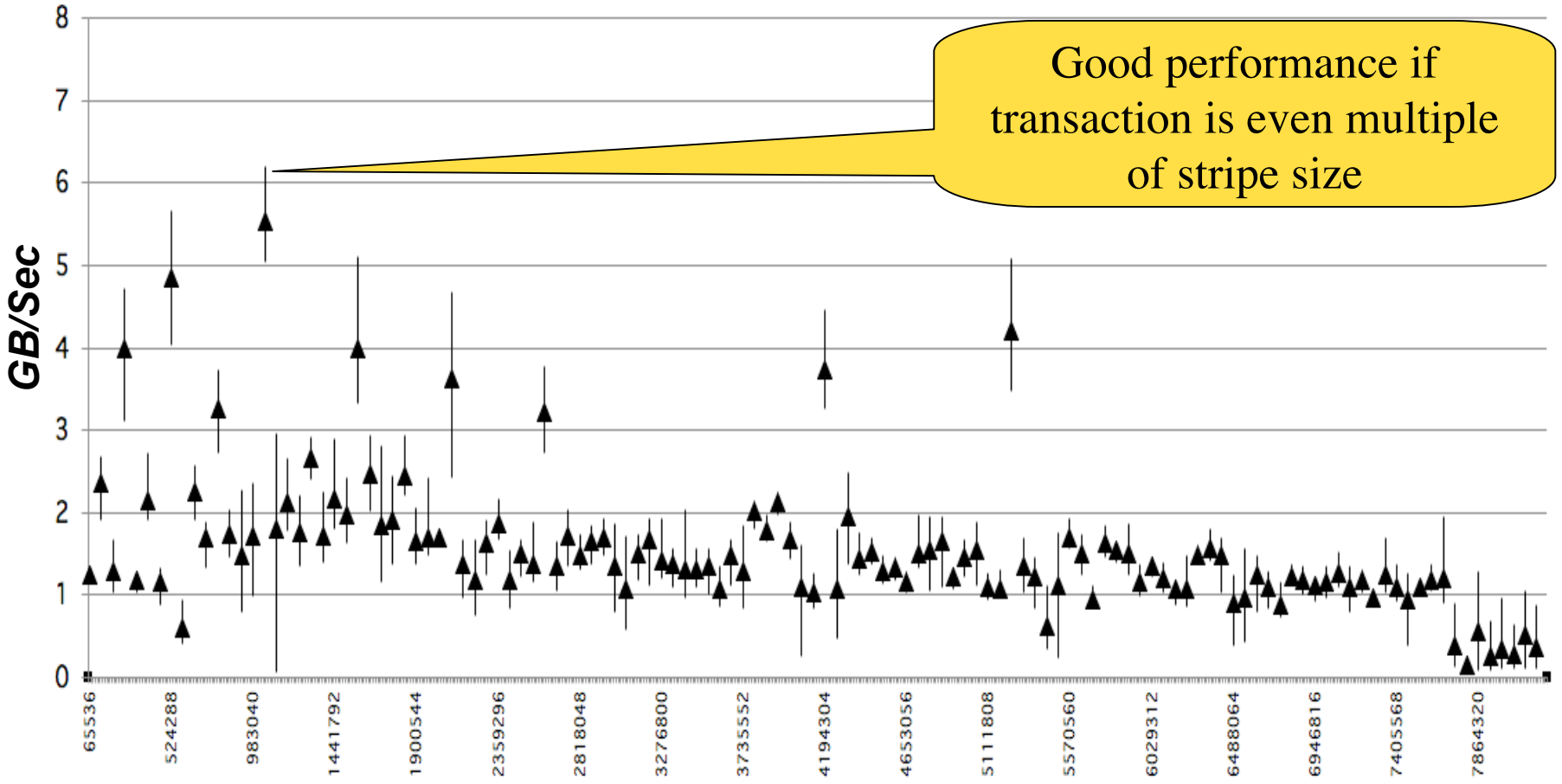
Decreased I/O Performance without Simple I/O Pattern

- Deviations from simple I/O patterns result in performance loss
  - Smaller amounts of data, (MBs/proc)
  - Transfer size not multiple of stripe width
  - Start offset doesn't match stripe width
  - Strided data



I/O Performance Sensitivity to Transfer Size

2GB File Size, 80 Processors, 40 OSTs

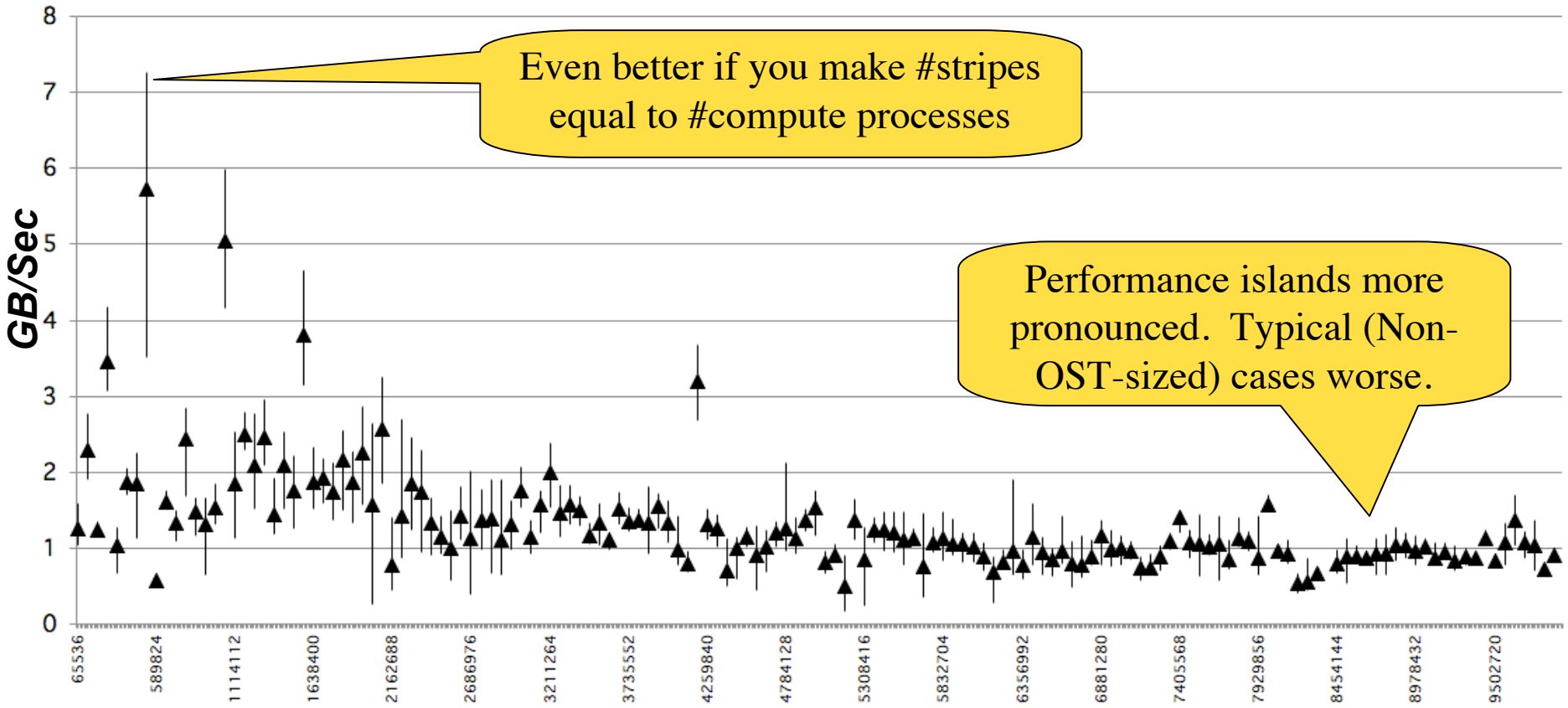


NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

Office of Science U.S. DEPARTMENT OF ENERGY

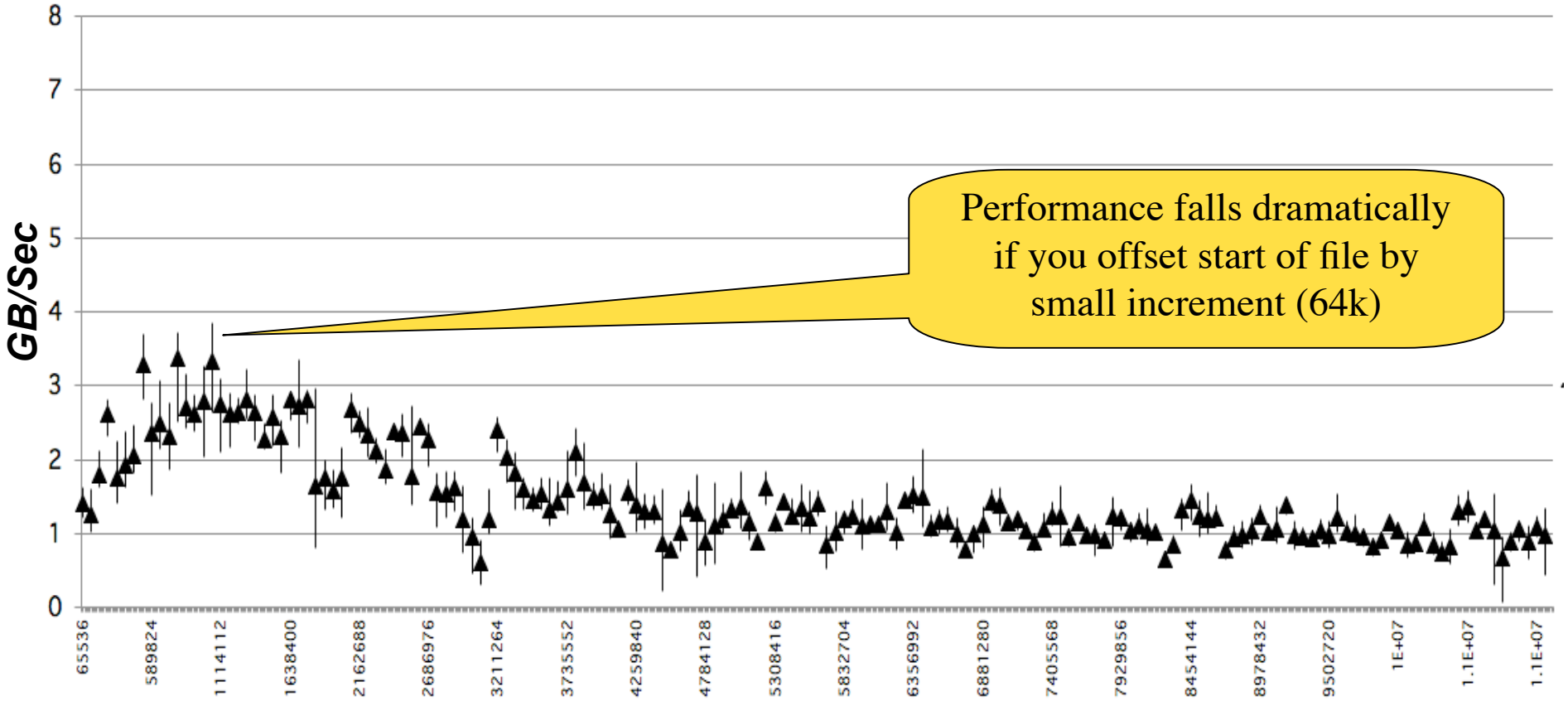
BERKELEY LAB

Sensitivity to Transfer Size with OST alignment



I/O Performance Sensitivity to Transfer Size

2GB File Size, 80 Processors 40 OSTs: Offset file start by 64k



NERSC NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

Office of Science U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

User Perspective: Impractical to aim for such small "performance islands"

- **Reasonable to help users adjust strategies at C/Fortran MPI, MPI-IO, HDF5 layer, but can't require users to understand low level file system details**
- **Transfer size for interleaved I/O must always match OST stripe width**
  - Difficult to constrain domain-decomposition to granularity of I/O
  - Not practical for codes which don't have identical domain sizes
- **Every compute node must write exactly aligned to OST boundary**
  - Not feasible if users write metadata or headers to files
  - Difficult for high-level self-describing file formats (HDF5, pnetcdf)
  - Not practical when domain-sizes are slightly non-uniform

Importance of MPI-IO Optimization for 2 Phase I/O

- We know:
  - Matching number of application I/O writers with number of OSTs gives best performance
  - Writing fewer, larger blocks of data gives better performance than many small writes
  - Writes aligning to OST boundaries get file per proc performance (impractical for apps)
- MPI-IO's 2 Phase I/O precisely addresses above concerns
  - Subset of nodes, called aggregators, do actual writing
  - Aggregators collect smaller chunks of data into larger blocks
  - More aggressive version addresses OST alignment

NERSC
NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

Priorities

- Improve MPI-IO implementation on Lustre
  - Investigate poor collective I/O performance
  - Support efforts of David Knaak to implement efficient 2 phase I/O including
    - Aligning data to block boundaries
    - Optimizations to match processors to OSTs
- HDF5 to take advantage of Lustre architecture and optimizations